A Distributed Algorithm for Large-Scale Generalized Matching

Faraz Makari, Baruch Awerbuch, Rainer Gemulla,
Rohit Khandekar, Julián Mestre, Mauro Sozio

max planck institut
informatik

Recommender systems

Given:

- A user-item feedback matrix

Goal:

- Recommend additional items users may like

Recommender systems

Given:

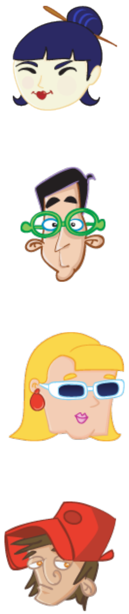
- A user-item feedback matrix

Goal:

- Recommend additional items users may like

Approach:

I. Predict missing ratings



4	1	2
5	1	3
4	4	2
3	5	2

Recommender systems

Given:

- A user-item feedback matrix

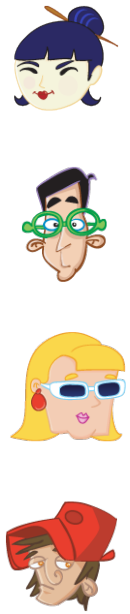
Goal:

- Recommend additional items users may like

Approach:

I. Predict missing ratings

II. Recommend items with highest predicted ratings



4	1	2
5	1	3
4	4	2
3	5	2

Recommender systems

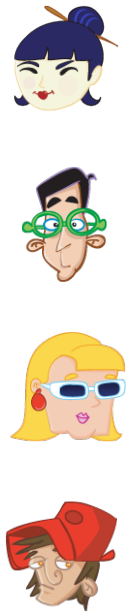
Given:
- A user-item feedback matrix

Goal:
- Recommend additional items users may like

Approach:

I. Predict missing ratings

II. Recommend items with highest predicted ratings



4	1	2
5	1	3
4	4	2
3	5	2

Recommender systems

Given:

- A user-item feedback matrix

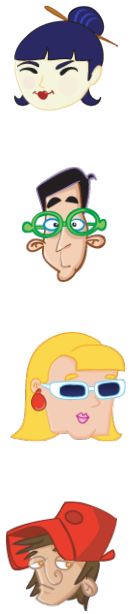
Goal:

- Recommend additional items users may like

Approach:

I. Predict missing ratings
II. Recommend items with highest predicted ratings

How to recommend items under constraints?



4	1	2
5	1	3
4	4	2
3	5	2

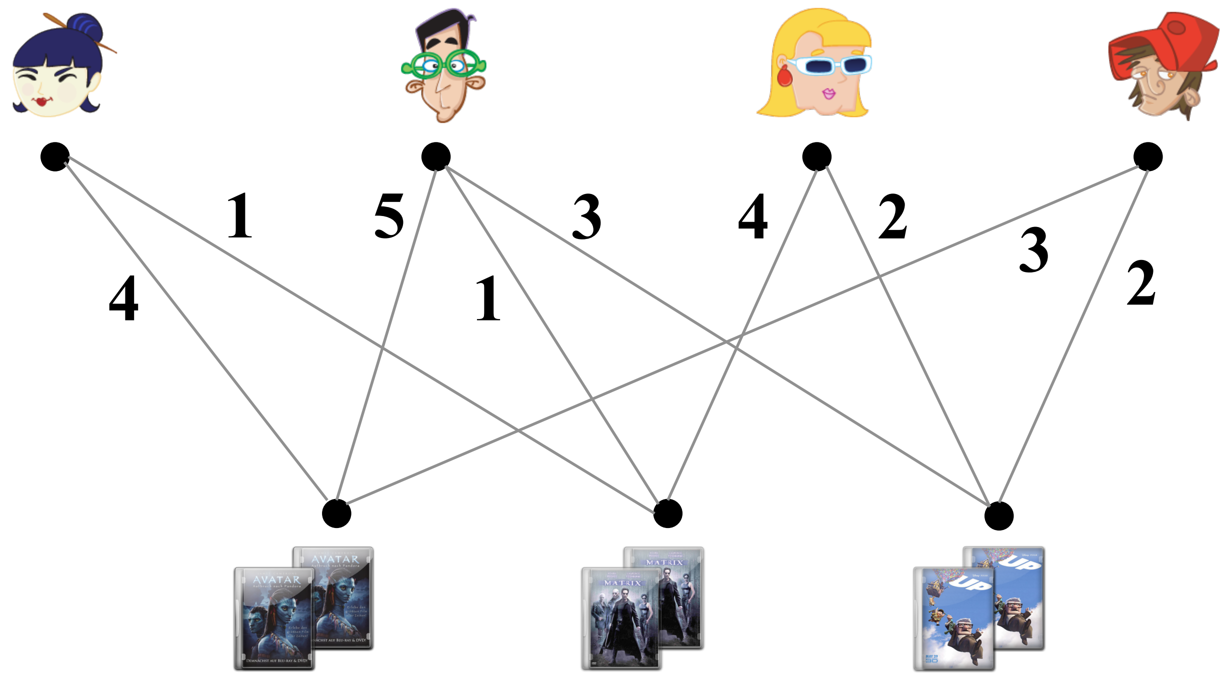
Generalized Bipartite Matching (GBM)

\# Recommendation = 1

Users

Predicted ratings

DVDs



Available DVD per movie = 2

Constraints:

- Neither too few nor too many recommendations
- Number of DVDs limited
- …

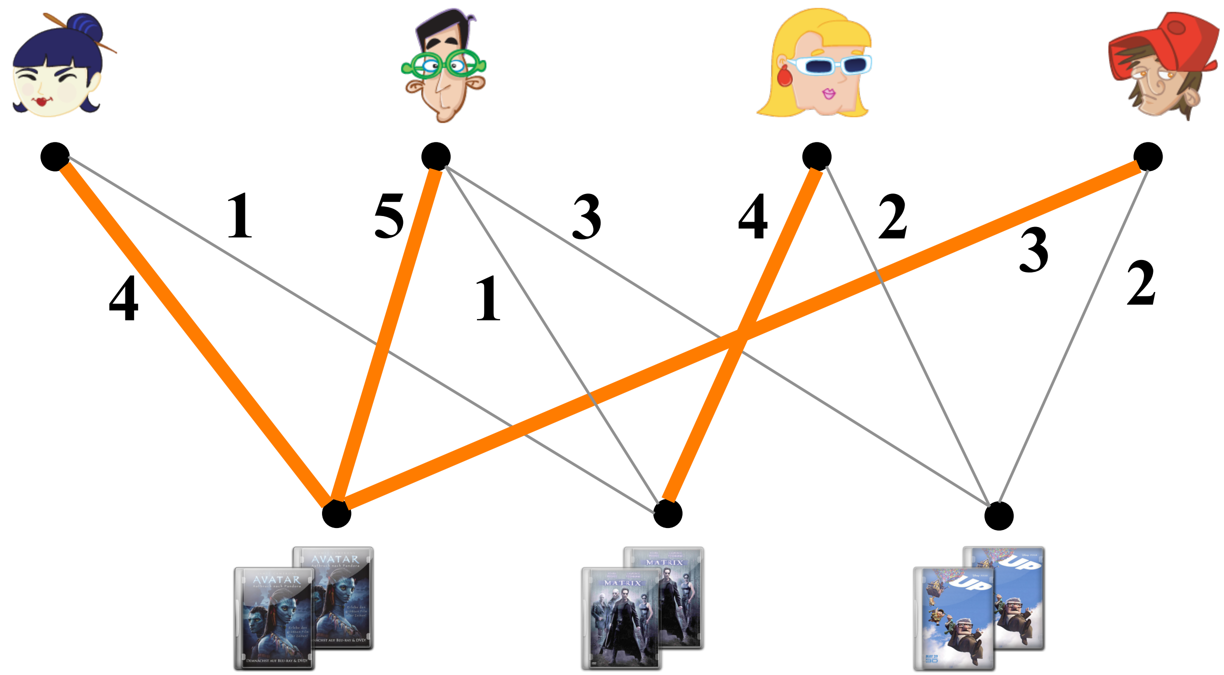
Generalized Bipartite Matching (GBM)

Recommendation = 1

Users

Predicted ratings

DVDs



Available DVD per movie = 2

Constraints:

- Neither too few nor too many recommendations
- Number of DVDs limited
- …

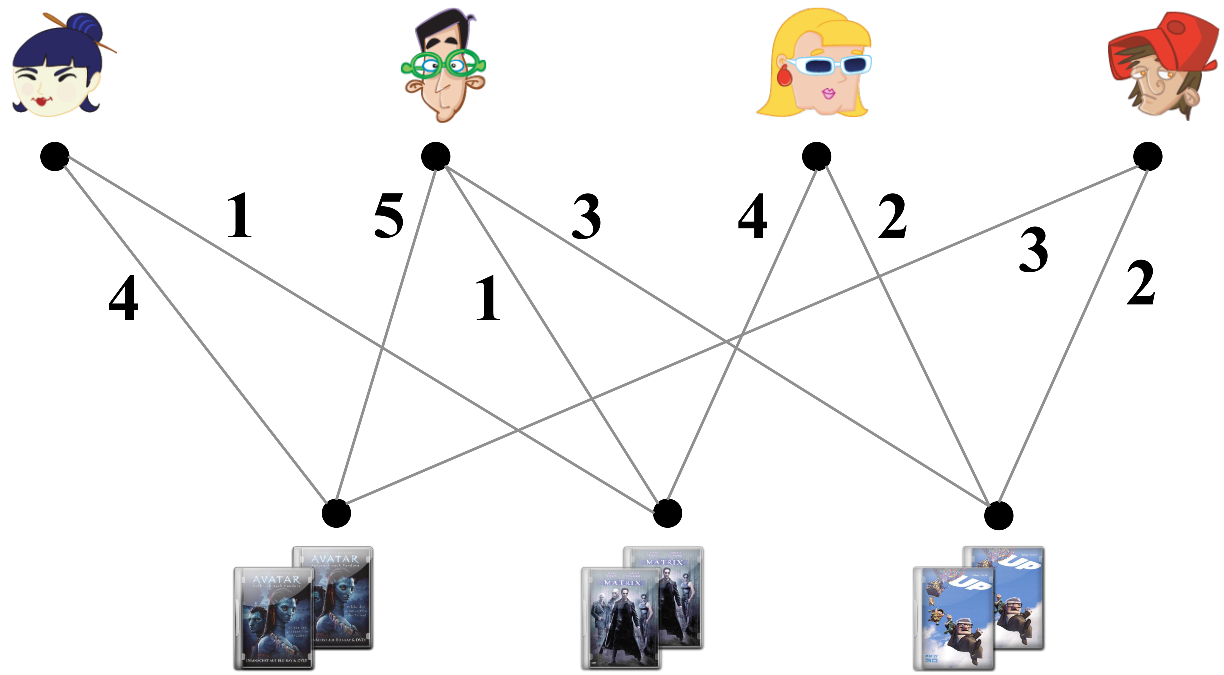
Generalized Bipartite Matching (GBM)

$1 \leq$ # Recommendations ≤ 2

Users

Predicted ratings

DVDs



Available DVD per movie = 2

Goal

- Recommend items to users s.t.
  - I. Interesting for users
  - II. Neither too few nor too many
  - III. Availability of items satisfied

s.t. Maximum weight matching (I); LB and UB constraints (II, III)

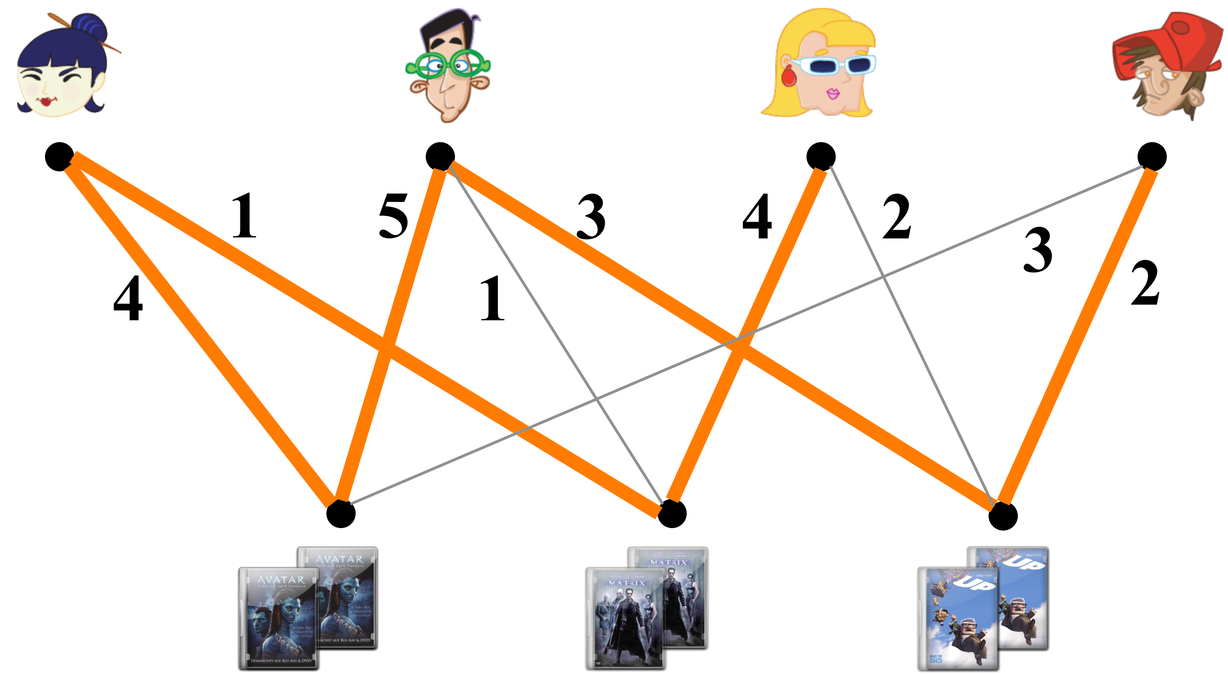
Generalized Bipartite Matching (GBM)

$1 \leq \#$ Recommendations ≤ 2

Users

Predicted ratings

DVDs



Available DVD per movie = 2

Goal

- Recommend items to users s.t.
  - I. Interesting for users
  - II. Neither too few nor too many
  - III. Availability of items satisfied

Maximum weight matching s.t. LB and UB constraints

Challenge

GBM optimally solvable in polynomial time

- E.g., linear programming
- Available solvers handle small instances very well

Challenge

GBM optimally solvable in polynomial time

- E.g., linear programming
- Available solvers handle small instances very well

Real applications can be large

- E.g., Netflix has >20M users, >20k movies
- Available solvers do not scale to large problems

Challenge

GBM optimally solvable in polynomial time

- E.g., linear programming
- Available solvers handle small instances very well

Real applications can be large

- E.g., Netflix has >20M users, >20k movies
- Available solvers do not scale to large problems

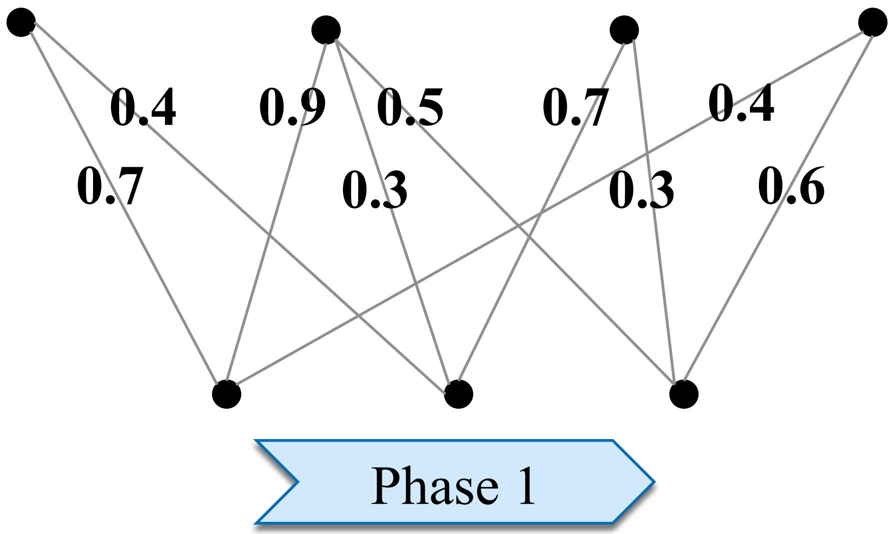
Goal:

- **Efficient** and **scalable** algorithm for large-scale GBM instances

Framework

Phase 1: Approximate LP

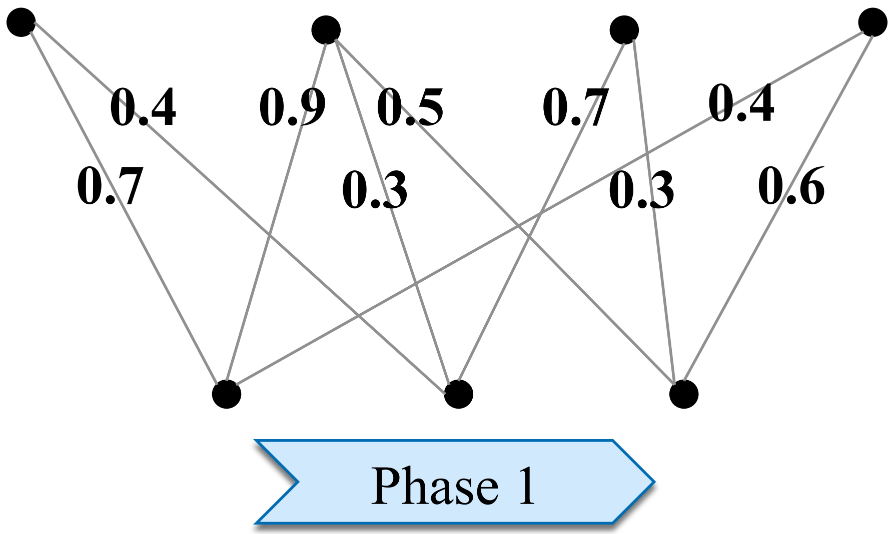
- Compute "edge probabilities" using linear programming



Framework

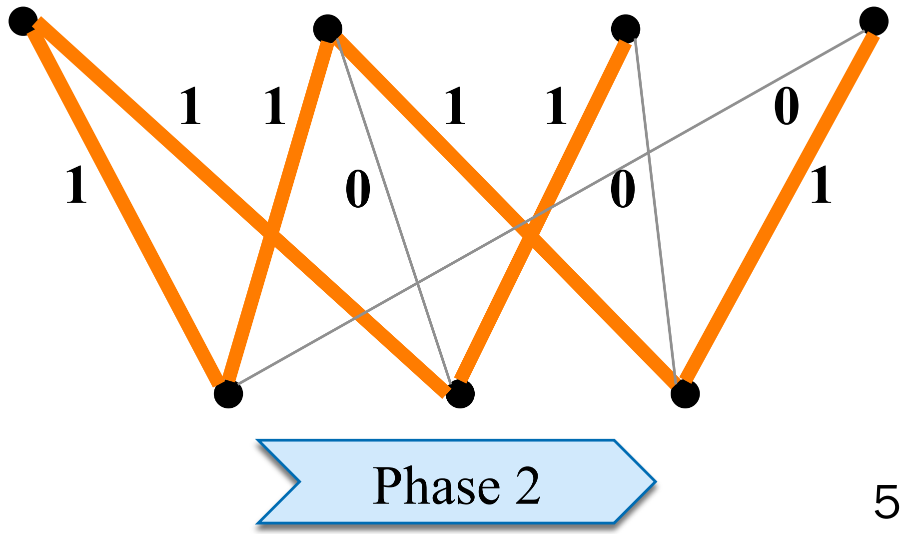
Phase 1: Approximate LP

- Compute "edge probabilities" using linear programming



Phase 2: Round

- Select edges based on probabilities from phase 1



Phase 1: Computing edge probabilities

Contrib. 1

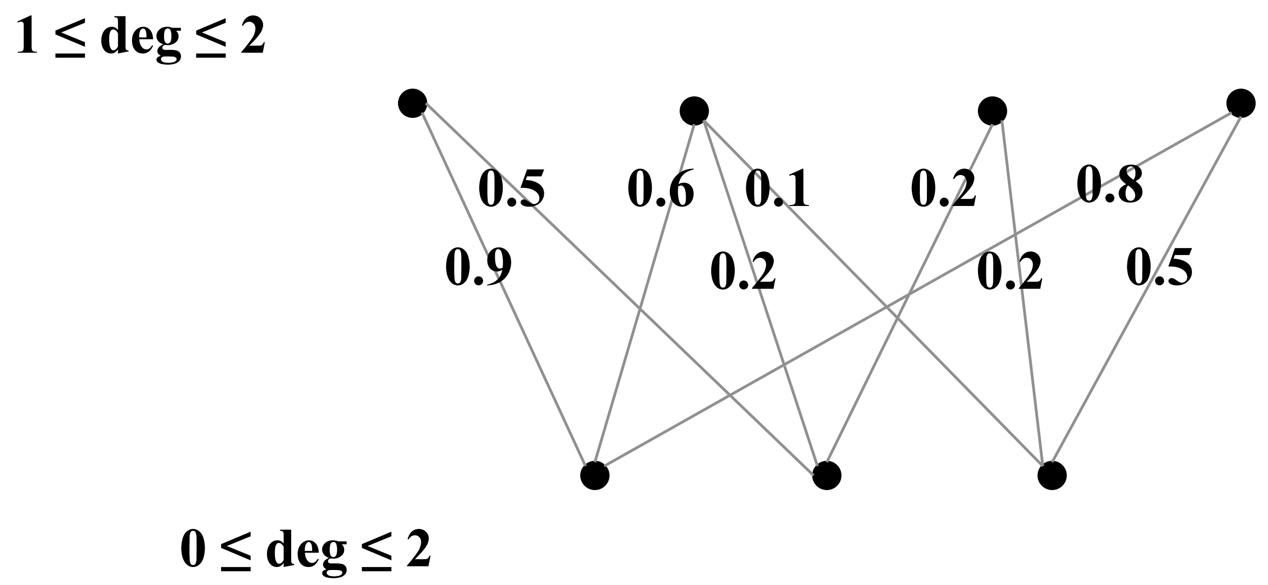
Algorithm for Mixed Packing Covering (MPC) LPs (like GBM LP)

- Gradient-based multiplicative weights update algorithm
- Approximately solves MPC LPs (ε: approx. parameter)
  - LB and UB constraints satisfied up to $(1\pm\varepsilon)$ ⟵ Almost feasible
  - Objective value $(1-\varepsilon)$ of the optimum ⟵ Near-optimal
- Poly-log rounds
- Easy to implement: Each round involves matrix-vector multiplications

Phase 1: Computing edge probabilities

Contrib. 1

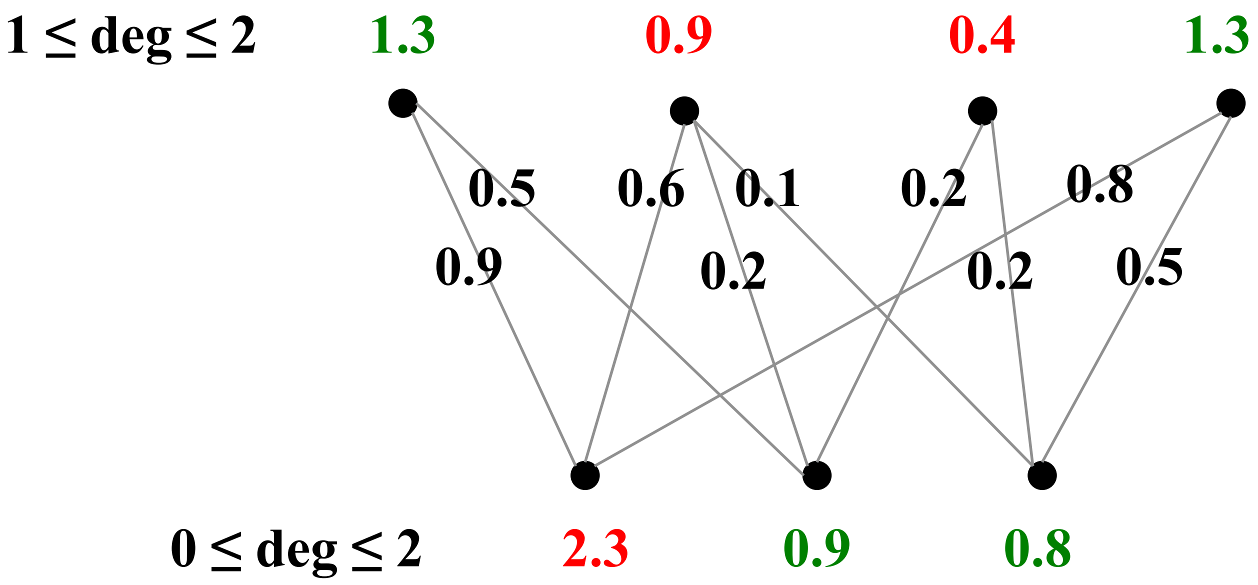
Algorithm for Mixed Packing Covering (MPC) LPs (like GBM LP)



Phase 1: Computing edge probabilities

Contrib. 1

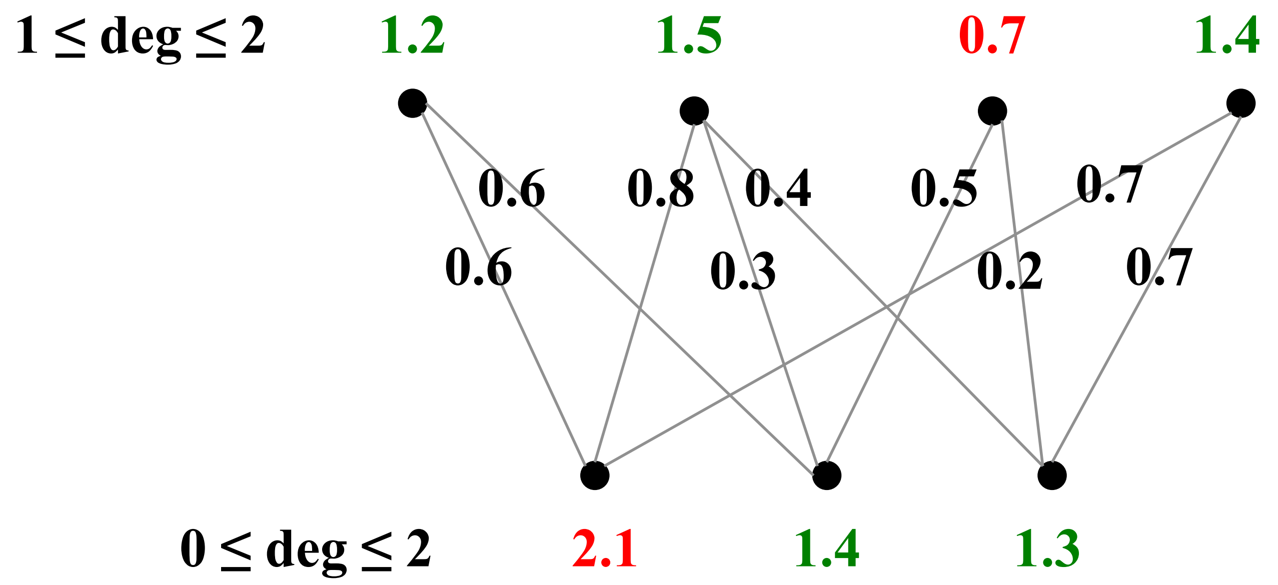
Algorithm for Mixed Packing Covering (MPC) LPs (like GBM LP)



Phase 1: Computing edge probabilities

Contrib. 1

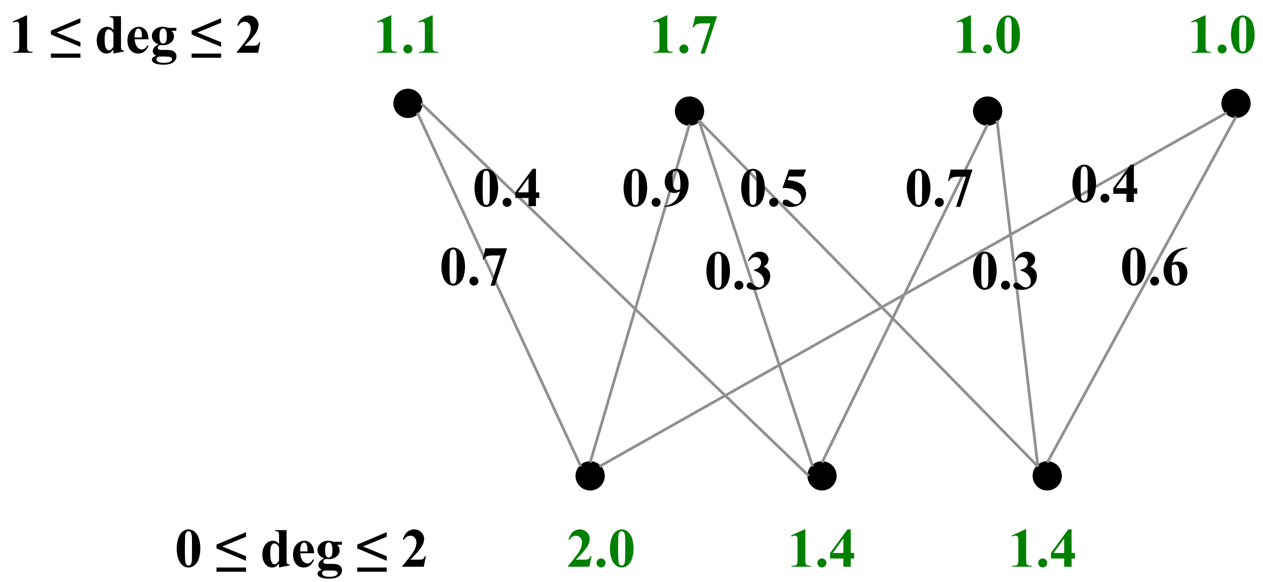
Algorithm for Mixed Packing Covering (MPC) LPs (like GBM LP)



Phase 1: Computing edge probabilities

Contrib. 1

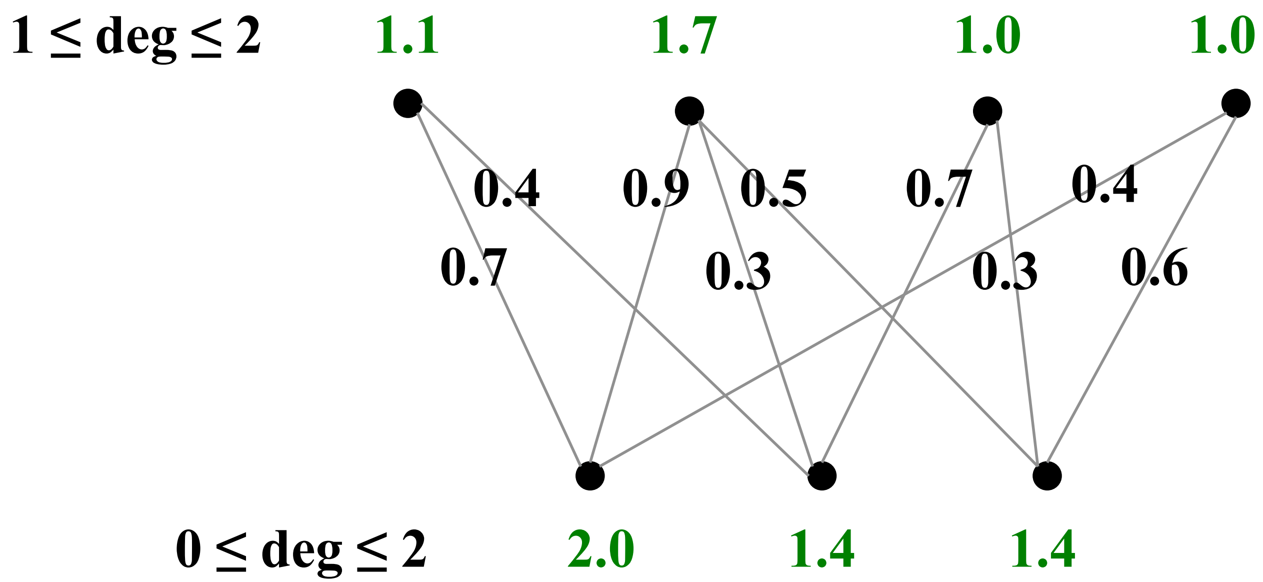
Algorithm for Mixed Packing Covering (MPC) LPs (like GBM LP)



Phase 1: Computing edge probabilities

Contrib. 1

Algorithm for Mixed Packing Covering (MPC) LPs (like GBM LP)



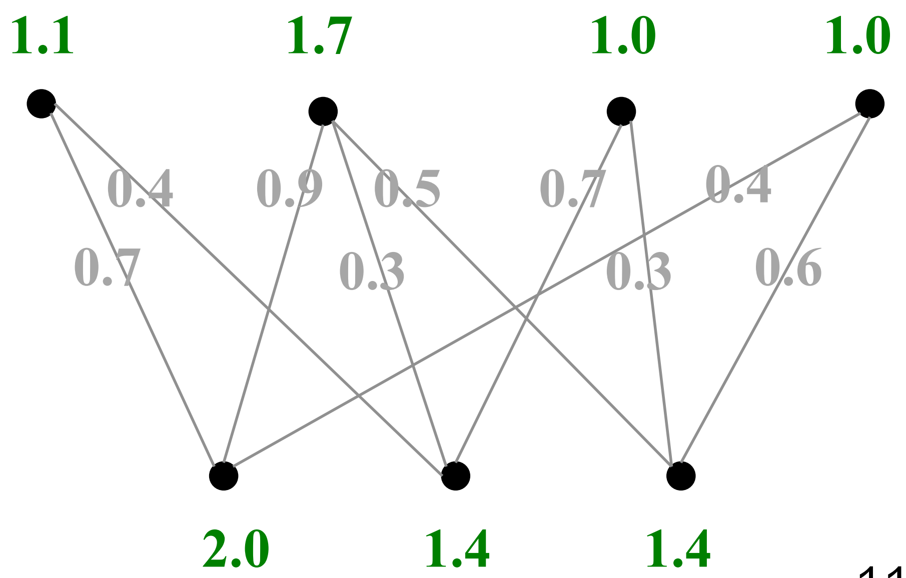
Contrib. 2

Distributed Processing for GBM (details in paper)

- Communication depends on # nodes not # edges
- All computation in parallel

Phase 2: Selecting edges

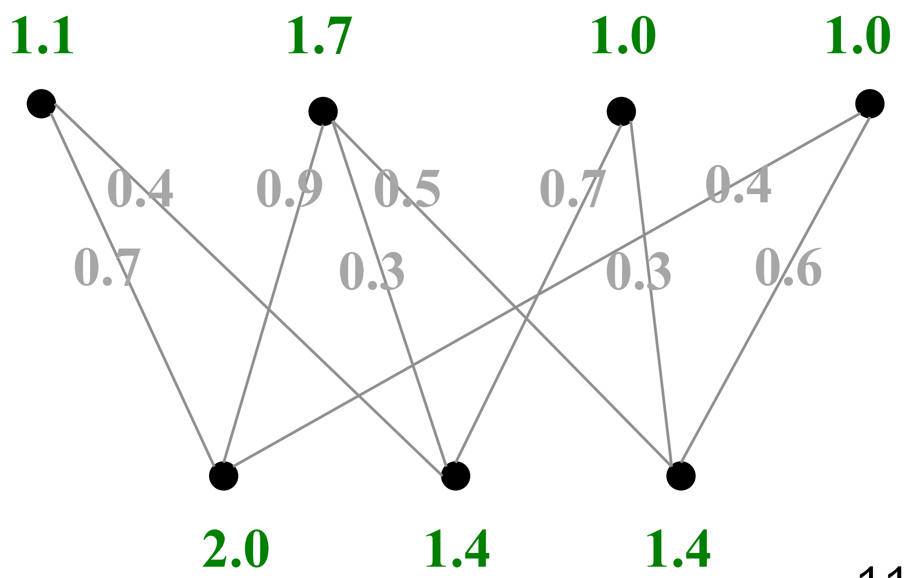
- **Given:** Edge probabilities from phase 1
- **Goal:** Select edges to be in final solution s. t.
  - I. LB and UB constraints satisfied (up to rounding)
  - II. Approx. guarantee preserved (in expectation)



Phase 2: Selecting edges

- **Given:** Edge probabilities from phase 1
- **Goal:** Select edges to be in final solution s. t.
  - I. LB and UB constraints satisfied (up to rounding)
  - II. Approx. guarantee preserved (in expectation)

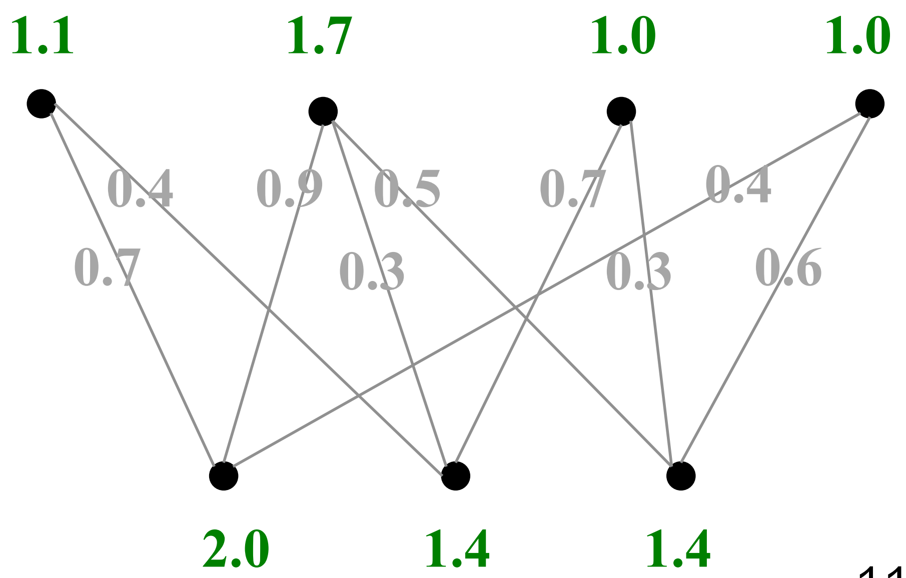
Naïve approach: Round independently using prob. from phase 1



Phase 2: Selecting edges

- **Given:** Edge probabilities from phase 1
- **Goal:** Select edges to be in final solution s. t.
  I. LB and UB constraints satisfied (up to rounding) ✖
  II. Approx. guarantee preserved (in expectation) ✔

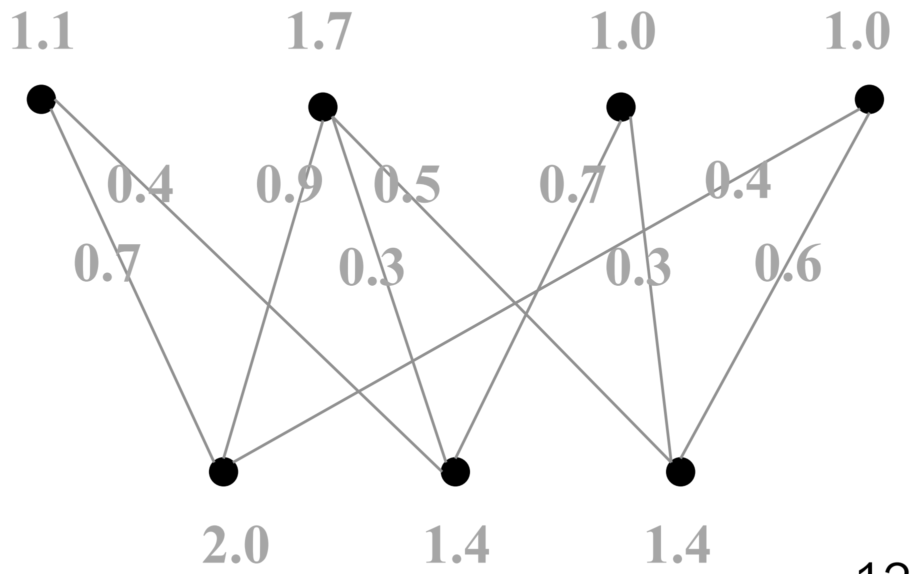
Naïve approach: Round independently using prob. from phase 1



Phase 2: Selecting edges

- **Given:** Edge probabilities from phase 1
- **Goal:** Select edges to be in final solution s. t.
  1. LB and UB constraints satisfied (up to rounding)
  2. Approx. guarantee preserved (in expectation)

Seq. algorithm [Gandhi et al. 06]:

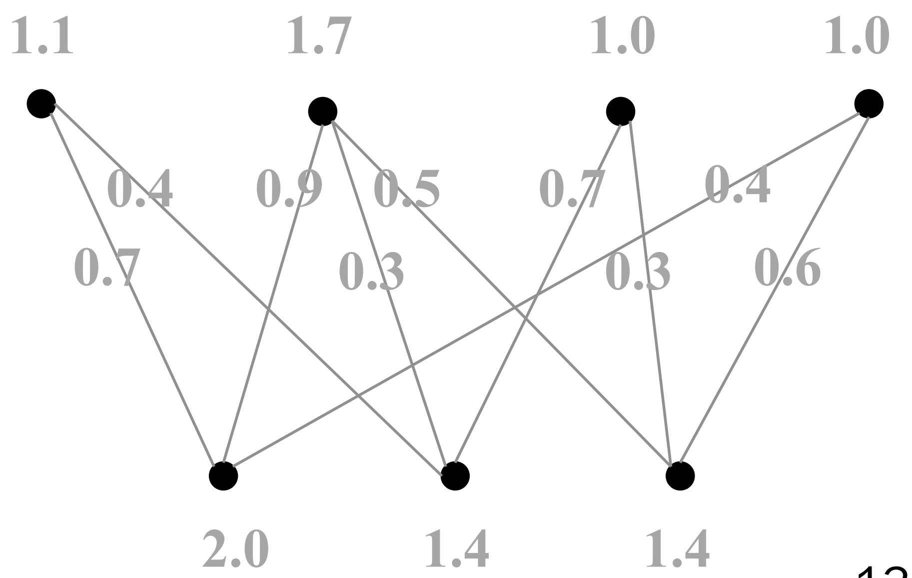


Phase 2: Selecting edges

- **Given:** Edge probabilities from phase 1
- **Goal:** Select edges to be in final solution s. t.
  - I. LB and UB constraints satisfied (up to rounding)
  - II. Approx. guarantee preserved (in expectation)

Seq. algorithm [Gandhi et al. 06]:

1. Find a cycle/maximal path

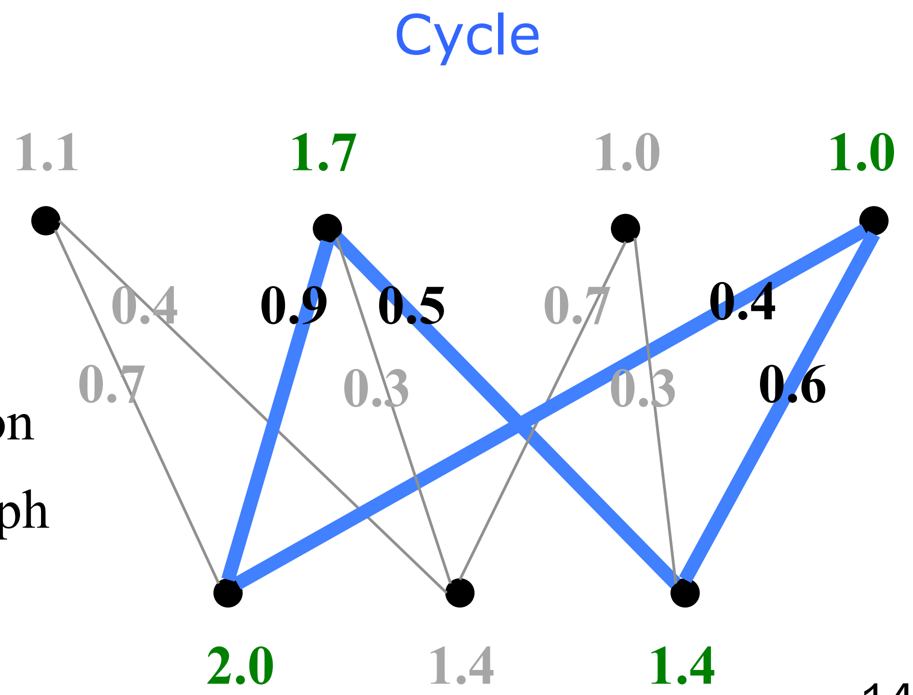


Phase 2: Selecting edges

- **Given:** Edge probabilities from phase 1
- **Goal:** Select edges to be in final solution s. t.
    - I. LB and UB constraints satisfied (up to rounding)
    - II. Approx. guarantee preserved (in expectation)

Seq. algorithm [Gandhi et al. 06]:

1. Find a cycle/maximal path
2. Modify edge prob. on the cycle/maximal path (details omitted)
    - If edge prob. = 1, include in solution
    - If edge prob. = 0, remove from graph

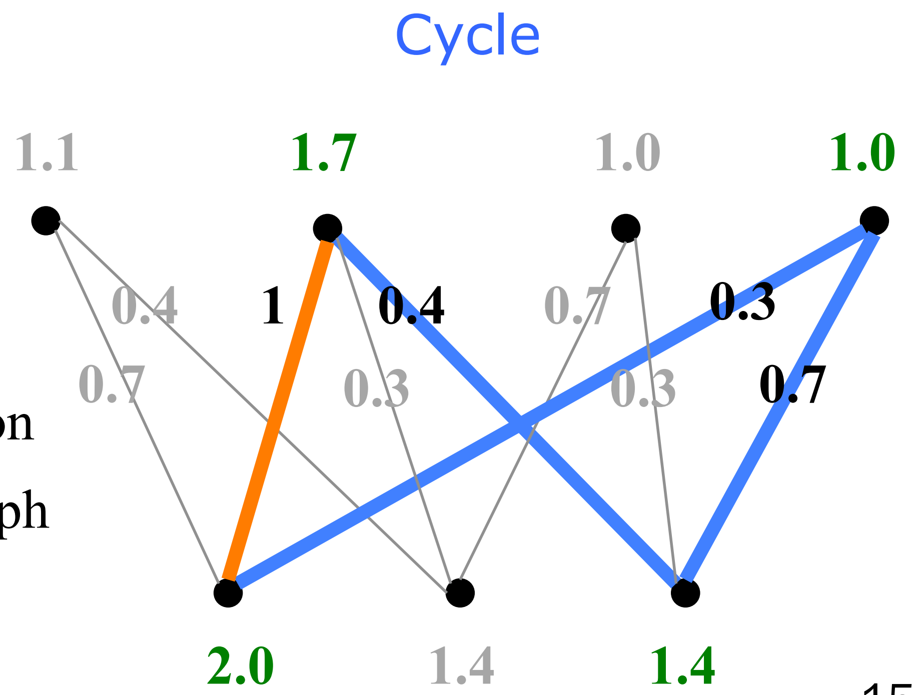


Phase 2: Selecting edges

- **Given:** Edge probabilities from phase 1
- **Goal:** Select edges to be in final solution s. t.
    I. LB and UB constraints satisfied (up to rounding)
    II. Approx. guarantee preserved (in expectation)

Seq. algorithm [Gandhi et al. 06]:

1. Find a cycle/maximal path
2. Modify edge prob. on the cycle/maximal path (details omitted)
    - If edge prob. = 1, include in solution
    - If edge prob. = 0, remove from graph



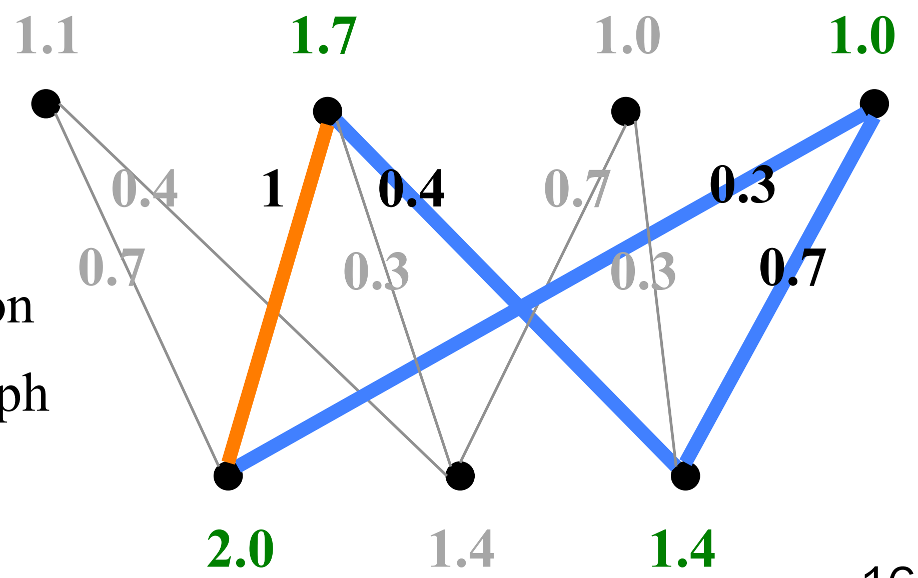
Phase 2: Selecting edges

- **Given:** Edge probabilities from phase 1
- **Goal:** Select edges to be in final solution s. t.
    I. LB and UB constraints satisfied (up to rounding)
    II. Approx. guarantee preserved (in expectation)

Seq. algorithm [Gandhi et al. 06]:

1. Find a cycle/maximal path
2. Modify edge prob. on the cycle/maximal path (details omitted)
    - If edge prob. = 1, include in solution
    - If edge prob. = 0, remove from graph
3. Repeat until graph is empty

Cycle

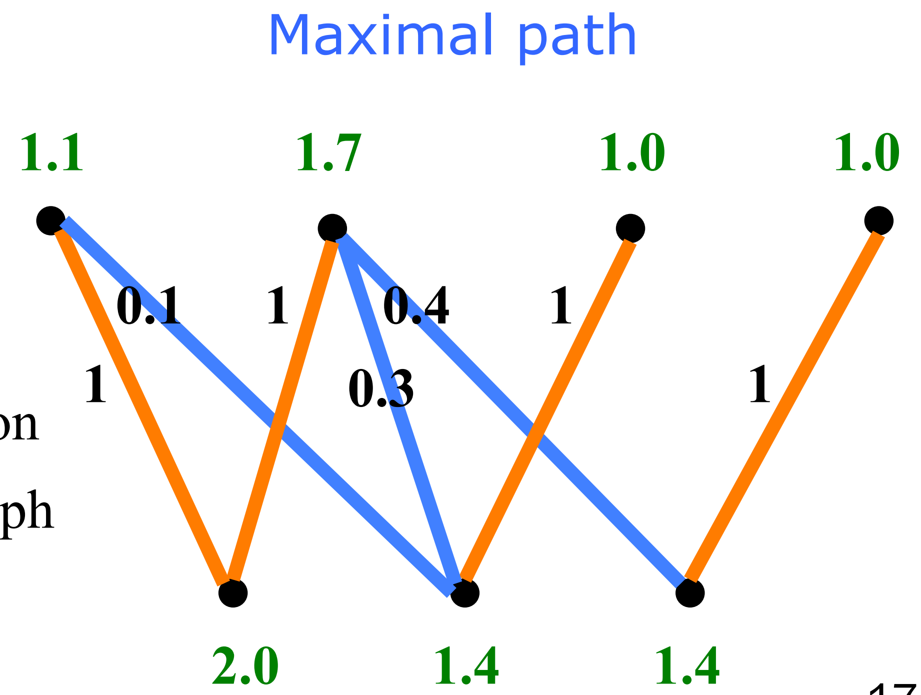


Phase 2: Selecting edges

- **Given:** Edge probabilities from phase 1
- **Goal:** Select edges to be in final solution s. t.
  - I. LB and UB constraints satisfied (up to rounding)
  - II. Approx. guarantee preserved (in expectation)

Seq. algorithm [Gandhi et al. 06]:

1. Find a cycle/maximal path
2. Modify edge prob. on the cycle/maximal path (details omitted)
   - If edge prob. = 1, include in solution
   - If edge prob. = 0, remove from graph
3. Repeat until graph is empty



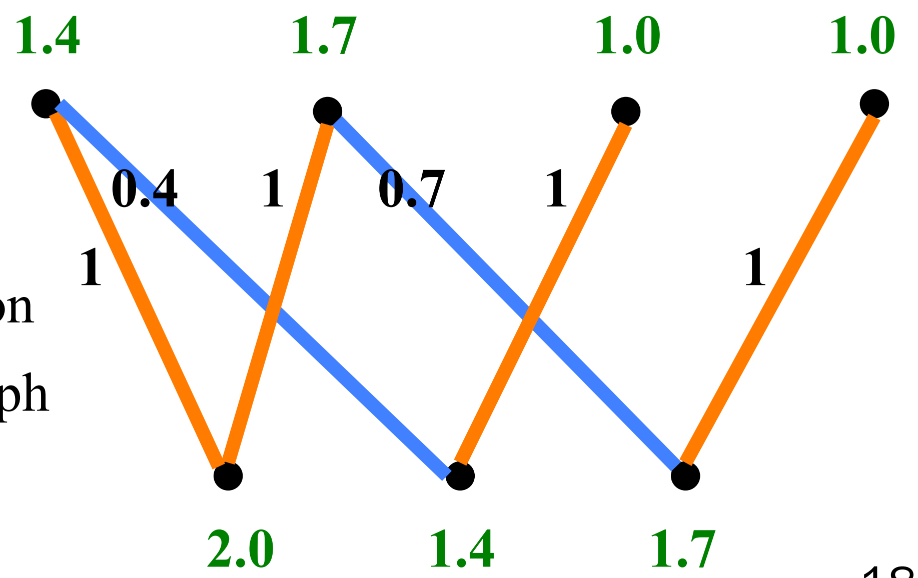
Phase 2: Selecting edges

- **Given:** Edge probabilities from phase 1
- **Goal:** Select edges to be in final solution s. t.
  - I. LB and UB constraints satisfied (up to rounding)
  - II. Approx. guarantee preserved (in expectation)

Seq. algorithm [Gandhi et al. 06]:

1. Find a cycle/maximal path
2. Modify edge prob. on the cycle/maximal path (details omitted)
   - If edge prob. = 1, include in solution
   - If edge prob. = 0, remove from graph
3. Repeat until graph is empty

Maximal path

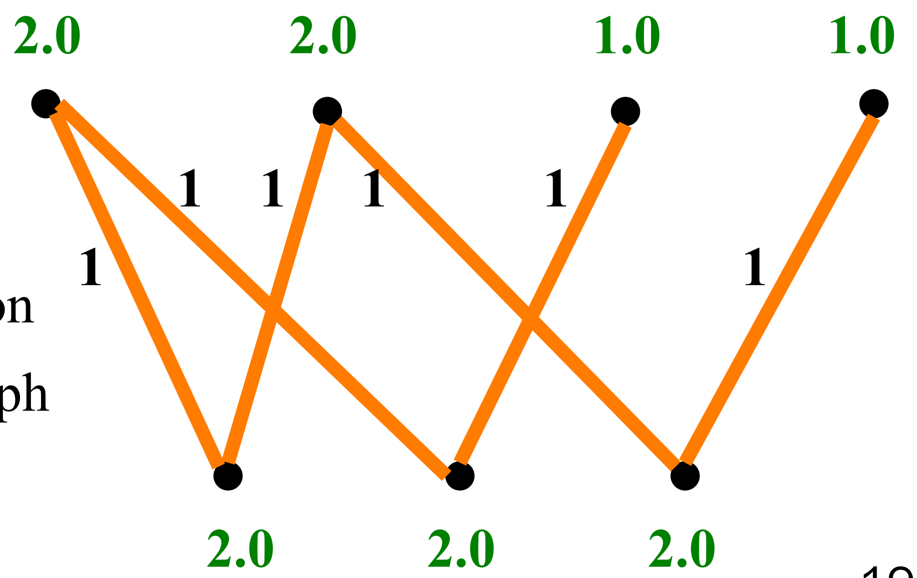


Phase 2: Selecting edges

- **Given:** Edge probabilities from phase 1
- **Goal:** Select edges to be in final solution s. t.
    - I. LB and UB constraints satisfied (up to rounding)
    - II. Approx. guarantee preserved (in expectation)

Seq. algorithm [Gandhi et al. 06]:

1. Find a cycle/maximal path
2. Modify edge prob. on the cycle/maximal path (details omitted)
    - If edge prob. = 1, include in solution
    - If edge prob. = 0, remove from graph
3. Repeat until graph is empty

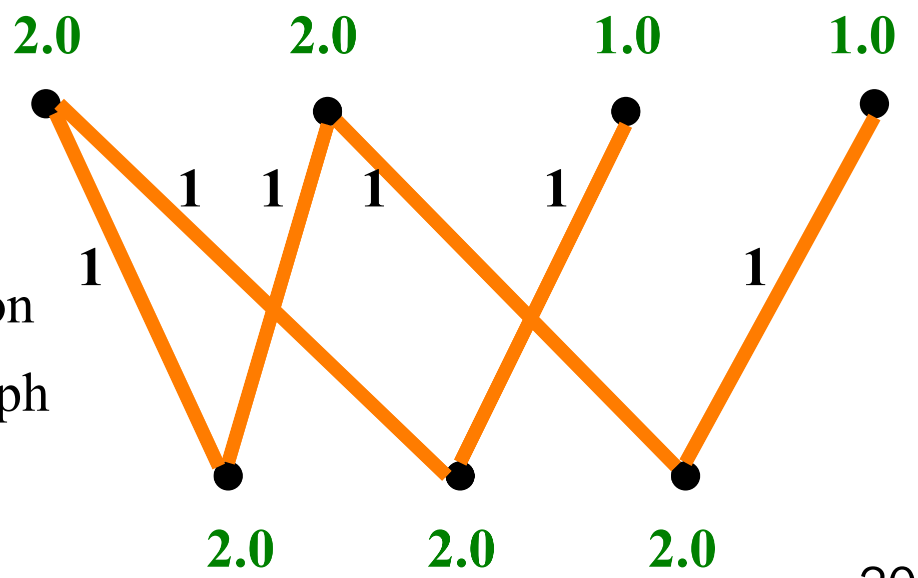


Phase 2: Selecting edges

- **Given:** Edge probabilities from phase 1
- **Goal:** Select edges to be in final solution s. t.
  - I. LB and UB constraints satisfied (up to rounding) ✔
  - II. Approx. guarantee preserved (in expectation) ✔

Seq. algorithm [Gandhi et al. 06]:

1. Find a cycle/maximal path
2. Modify edge prob. on the cycle/maximal path (details omitted)
   - If edge prob. = 1, include in solution
   - If edge prob. = 0, remove from graph
3. Repeat until graph is empty

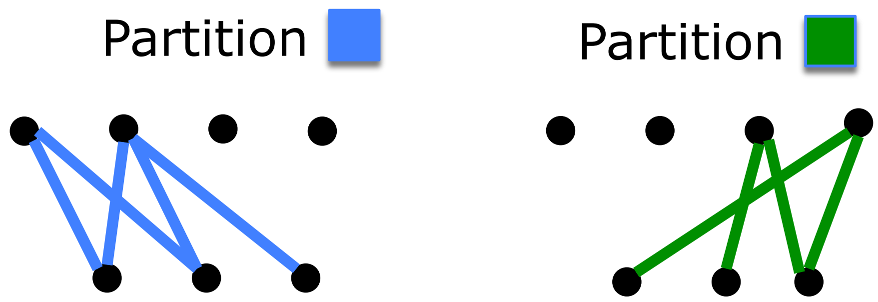


Phase 2: Selecting edges

Contrib. 3

How to distribute?

- A local cycle is a global cycle
- A local maximal path is not a global maximal path
- Order of processing cycles has no affect on approx. guarantess

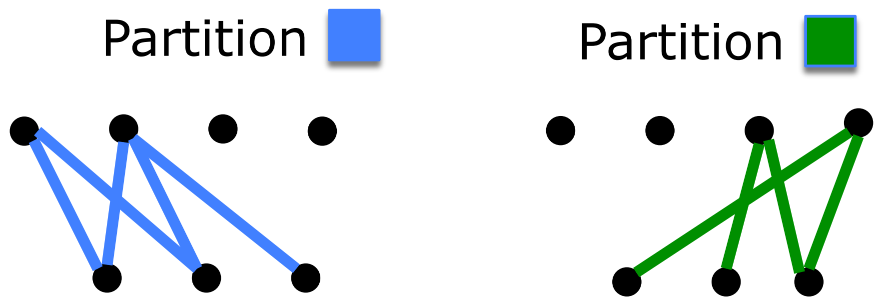


Phase 2: Selecting edges

Contrib. 3

Distributed algorithm:

1. Partition edges uniformly across compute nodes
2. Process **all cycles** in each partition
3. Merge all partitions
4. Repeat until graph is “small enough”
5. On last partition: Process **all cycles and maximal paths** using seq. alg.

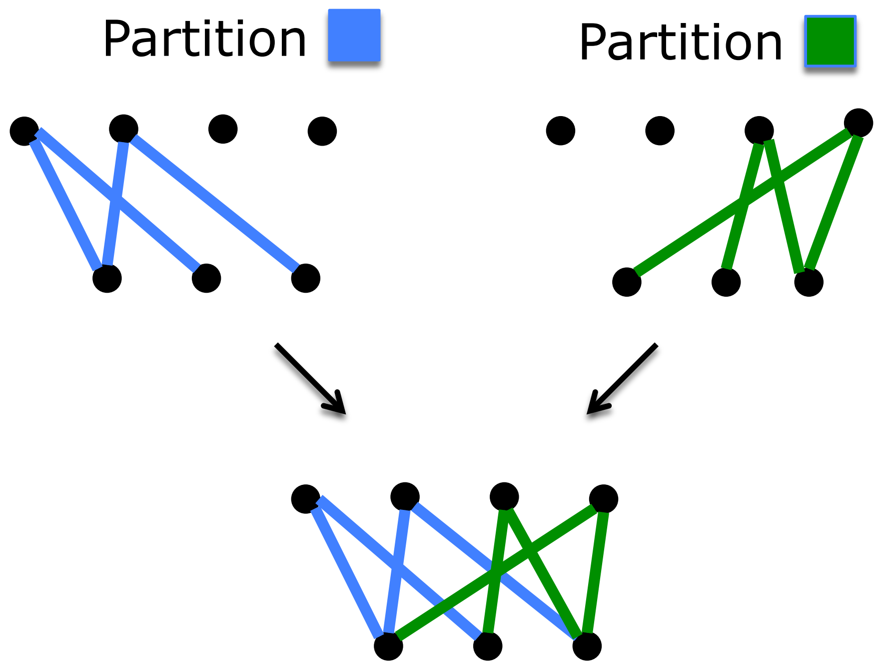


Phase 2: Selecting edges

Contrib. 3

Distributed algorithm:

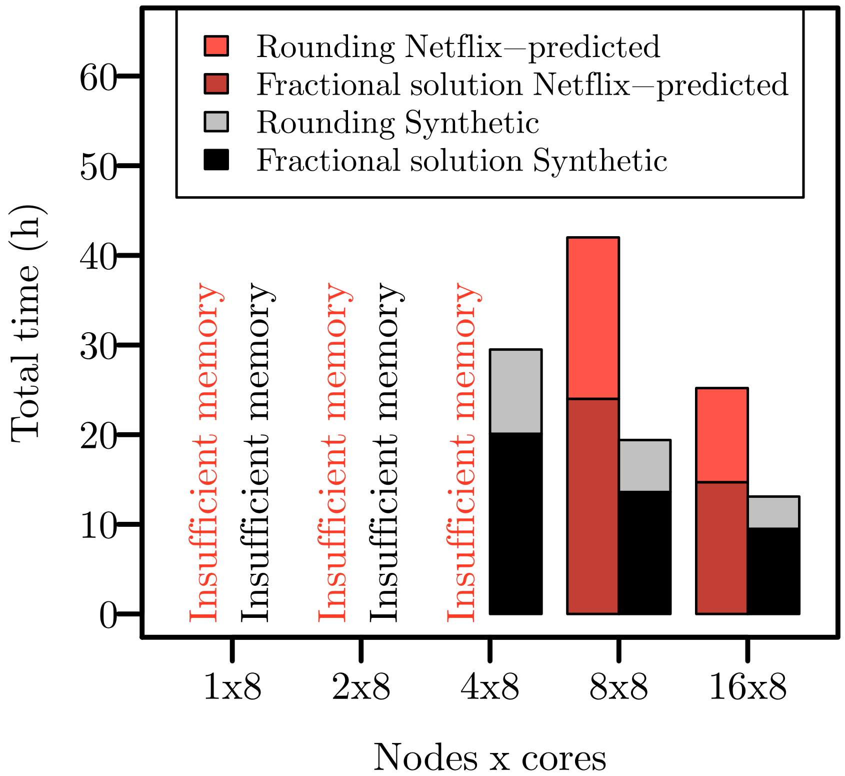
1. Partition edges uniformly across compute nodes
2. Process **all cycles** in each partition
3. Merge all partitions
4. Repeat until graph is “small enough”
5. On last partition: Process **all cycles and maximal paths** using seq. alg.



Large-Scale (Semi-) Synthetic Experiments

Dataset	# users	# items	# edges
Netflix-predicted	490k	18k	3.2B
Synthetic	10M	1M	1B

Cluster of commodity nodes



Powerful server

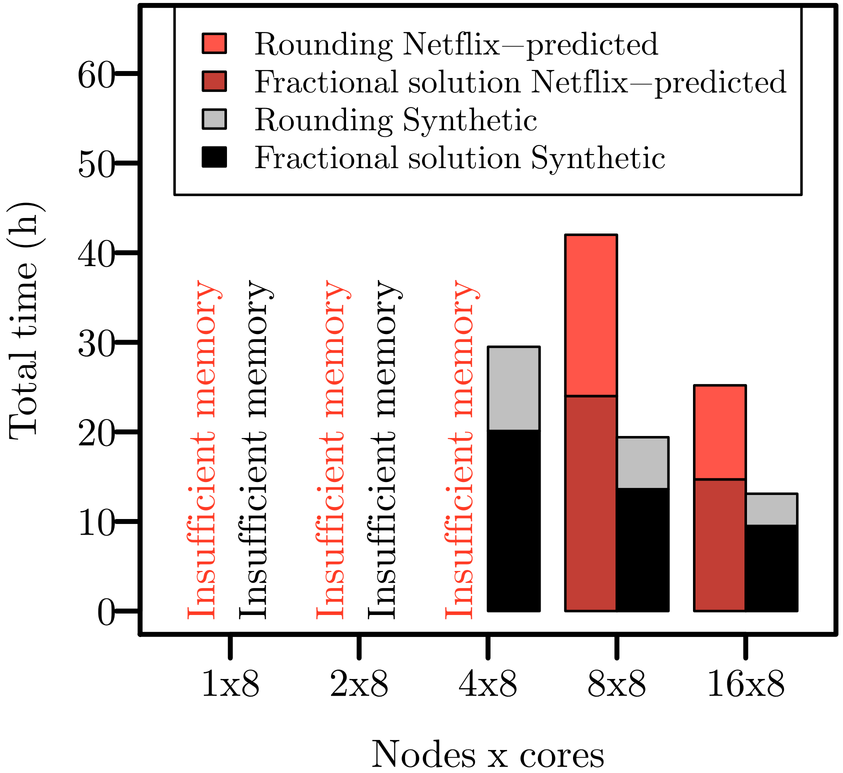
Gurobi ran out of memory on a large-memory server with 512GB RAM

Large-Scale (Semi-) Synthetic Experiments

Scales to very large problem instances

Dataset	# users	# items	# edges
Netflix-predicted	490k	18k	3.2B
Synthetic	10M	1M	1B

Cluster of commodity nodes



Powerful server

Gurobi ran out of memory on a large-memory server with 512GB RAM

Summary

- Recommending items to users under constraints
- Contributions:
  - A scalable distributed algorithm for large-scale GBM
  - A simple and efficient algorithm for general MPC LPs
  - Effective distributed processing for GBM
  - A distributed randomized rounding for GBM
- Experiments indicate scalability to instances with millions and nodes and billions of edges

Summary

- Recommending items to users under constraints
- Contributions:
  - A scalable distributed algorithm for large-scale GBM
  - A simple and efficient algorithm for general MPC LPs
  - Effective distributed processing for GBM
  - A distributed randomized rounding for GBM
- Experiments indicate scalability to instances with millions and nodes and billions of edges

Thank you
Questions?